📌Décisions Techniques - Catalogue Produit Modulaire

Q1 : Scalabilité & Performance

Comment optimiser /products/search et /products/trending pour >100K produits ?

🔍/products/search

- ✅**Indexation MongoDB** : indexes textuels sur `name`, `description`, `tags`.
- ✅**Filtrage par MongoDB** directement : pas de traitement en mémoire.
- ✅**Pagination efficace** : `limit` + `skip` ou par curseur (`_id`, `createdAt`).
- ✅**Cache Redis** : pour les requêtes populaires.
- ✅**ElasticSearch (optionnel)** : pour une recherche avancée.

🗜️/products/trending

- ✅**Pré-calcul d'un champ popularityScore** dans un cron job.
- ✅**Tracking Redis des vues/commandes** avec flush périodique en DB.
- ✅**Index sur popularityScore** pour tri rapide.

Q2 : Sécurité

🔐Comment protéger les endpoints de rafraîchissement de token ?

- ✅Stockage en **cookie HTTP-only**
- ✅**Vérification en DB/Redis** du token
- ✅**Rotation du refresh token**
- ✅**Rate limiting** + journaux d'anomalies

🛡️Upload de fichiers

- ✅Validation MIME : jpg, png, gltf, etc.
- ✅Renommage automatique du fichier
- ✅Taille max avec multer (`limits.fileSize`)
- ✅Aucun exécutable, pas de stockage dans un dossier exécutable

---

Q3 : Internationalisation

Comment gérer le multilingue pour `` ?

- ✅Stocker les champs comme objets multilingues :

```
name: {
  fr: "Sac à dos",
  en: "Backpack",
```

```
  ar: "حقيبة"
}
```

- ✅Middleware pour détecter `lang` (query ou header)
- ✅Le backend retourne uniquement la langue demandée
- ✅Possibilité de fallback (ex: `en` si `fr` indisponible)

Q4 : Analytics

Comment suivre les vues, tendances et ratings produits ?

🔍Vues produit

- ✅Incrémenter dans Redis : `INCR product:123:views`
- ✅Flush régulier vers MongoDB avec cron

📊Tendances

- ✅Score calculé : `views * 0.4 + achats * 0.4 + rating * 0.2`
- ✅Mis à jour par cron, stocké dans `popularityScore`

🕐Ratings

- ✅Recalcul dynamique à chaque nouvel avis
- ✅Ou maintenir `ratingSum` et `numReviews`
- ✅ `/rating-breakdown` via `$group` Mongo ou champ pré-calculé

Q5 : Stack technique choisie

Domaine	Stack choisi	Justification
Backend Framework	**Express.js**	Minimaliste, performant, facile à moduler pour API REST
Frontend Framework	**React.js**	Composants réutilisables, dashboard dynamique
Base de données	**MongoDB**	NoSQL, structure flexible pour produits/ variantes/reviews
Stockage de fichiers	**Local avec Multer**	Simple, efficace pour `/uploads`, sans coûts cloud
Authentification	**JWT + bcrypt + helmet + cors**	Stateles, chiffrage des mots de passe, middleware sécurité
DevOps	**GitHub + GitHub Actions**	CI/CD intégré, tests automatiques, déploiement rapide
Validation API	**express-validator**	Validation robuste des entrées côté serveur

Domaine	Stack choisi	Justification
Docs API	**Swagger + Postman Collection**	Documentation interactive et collection de tests manuels
Upload Middleware	**Multer (stockage local)**	Contrôle du type, taille, et destination locale